Fast Fingerprint Classification with Deep Neural Networks

Michelsanti, Daniel; Ene, Andreea-Daniela; Guichi, Yanis; Stef, Rares; Nasrollahi, Kamal; Moeslund, Thomas B.

Published in:
VISiGRAPP 2017

DOI (link to publication from Publisher):
10.5220/0006116502020209

Publication date:
2017

Document Version
Accepted author manuscript, peer reviewed version

Fast Fingerprint Classification with Deep Neural Networks

Keywords: Fingerprint Classification, Transfer Learning, Convolutional Neural Networks.

Abstract: Reducing the number of comparisons in automated fingerprint identification systems is essential when dealing with a large database. Fingerprint classification allows to achieve this goal by dividing fingerprints into several categories, but it presents still some challenges due to the large intra-class variations and the small inter-class variations. The vast majority of the previous methods uses global characteristics, in particular the orientation image, as features of a classifier. This makes the feature extraction stage highly dependent on preprocessing techniques and usually computationally expensive. In this work we evaluate the performance of two pre-trained convolutional neural networks fine-tuned on the NIST SD4 benchmark database. The obtained results show that this approach is comparable with other results in the literature, with the advantage of a fast feature extraction stage.

1 INTRODUCTION

Physiological and behavioural characteristics have always been used to identify an individual. When referring to this identification approach, the focus is on biometrics, which includes traits such as fingerprint, face, and voice. Lately, because of the progress in computer processing, automatic biometric systems, based on concepts developed long ago, have become available (Mayhew, 2015).

Today, biometrics are generally preferred to the traditional token-based systems, such as identity cards, driver's licences, keys, or knowledge-based systems, such as passwords. The reason is that the latter identification methods might be easily forgotten or shared, which has a negative impact on their reliability. The recent increase of fingerprint sensors adopted in smartphones for login and payment security systems, is only one example of this trend.

When a person requires to be identified through his/her fingerprint, it is necessary to compare that trait with the entire set of fingerprints in a database (Maltoni et al., 2009). If the database is large, this approach may be problematic for real-time applications due to the high number of comparisons needed. For this reason, fingerprints are often divided into different classes. Generally, four classes are used: Arch (A), Left loop (L), Right loop (R), Whorl (W). Sometimes the first class is divided into two categories, Arch (A) and Tented arch (T), as shown in Figure 1. In this paper, the four-class classification problem is taken into consideration.

The existing methods for fingerprint classification are based on classifiers that use hand-crafted features extracted from fingerprint images (Maltoni et al., 2009). These classification systems are highly dependent on a preprocessing phase that increases the processing time at test stage. Motivated by the recent success of deep learning techniques in many computer vision tasks, the proposed system in this paper consists of a pre-trained Convolutional Neural Network (CNN) fine-tuned on a subset of the NIST SD4 benchmark database (Watson and Wilson, 1992a). The two used CNN architectures are VGG-F and VGG-S (Chatfield et al., 2014). The main contribution of this work is to show the performance of transfer learning for a problem which is generally tackled by extracting visually meaningful features.

The paper is structured as follows: in Section 2 the state-of-the art is detailed. Section 3 provides a description of the database used in this work, and Section 4 shows the architecture of the adopted networks, the parameters for training, and the evaluation protocol. Finally, the results are presented and discussed, and the conclusions are drawn.

2 PREVIOUS WORK

2.1 Fingerprint Classification

Fingerprint classification is a pattern recognition problem that has received considerable attention for its difficulty, due to the small inter-class variability and the large intra-class variability (Maltoni et al., 2009). A list of the most relevant works can be found in (Galar et al., 2015).

When referring to fingerprints, images show three kinds of features based on the level of ridge details

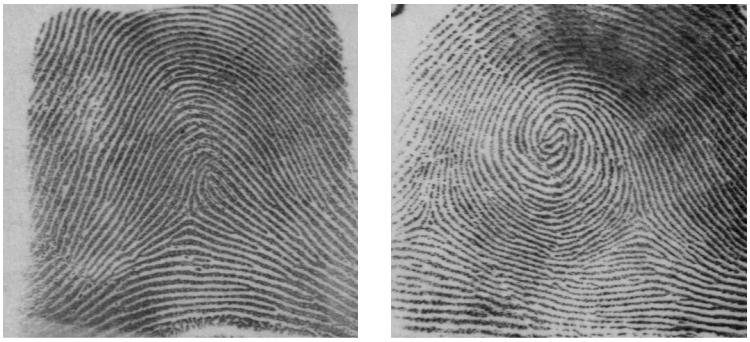

Figure 1: Fingerprint classes.

(Maltoni et al., 2009):

- At the first level (global) we have three possible singular regions: loop, delta, and whorl.
- The second level (local) allows us to find small details in fingerprint patterns, known as minutiae.
- The third level (fine-detail) shows all the attributes of the ridges.

Although some approaches that use feature-learning algorithms have been proposed in the past, such as (Tan et al., 2003), global characteristics are generally used for classification, in particular: ridge line flow, orientation image, singular points, and Gabor filter responses (Maltoni et al., 2009). Among them, the orientation image is the most used.

The feature extraction can be performed in several ways, depending on the kind of feature, the quality of the image, and the accuracy that the following classification phase requires. The goal of the classification stage is to learn a classifier based on labelled fingerprints. The techniques proposed in the literature are evaluated on a number of different databases (Galar et al., 2015), like NIST SD4, NIST SD14 (Watson and Wilson, 1993), NIST SD9 (Watson and Wilson, 1992b), with the first one as the most popular one. They can be divided into these categories (Maltoni et al., 2009):

- Rule-based - The classification is made according to the number of singularities:
  - Arch has no singular points.
  - Tented arch, and left/right loop have one loop and one delta.
  - Whorl has two loops or a whorl, and two deltas.
- Syntactic - These methods are based on a grammar defined by symbols extracted from the features of the fingerprints.
- Structural - It includes methods that build data structures, such as trees or graphs, for a better relational organisation of low-level features into higher-level hierarchical structures.
- Statistical - In this case, the classifier is a statistical one, like Bayesian Decision Rule, K-Nearest Neighbour, and Support Vector Machine (SVM).
- Neural network-based - Generally the classifier consists of a Multi Layer Perceptron, after the dimensionality reduction performed on the feature vector.

- Multi-classifier - This category includes all the approaches that combine two or more classifiers.

Our work differs from the previous ones because it is not based on global features, but it performs the feature extraction and the classification through a CNN.

2.2 Transfer Learning

The main problem of using CNNs is the long time and the huge size of the dataset needed for training. A solution is to use a network pre-trained on a different dataset. There are two possibilities:

1. Use the network to extract features and then train a linear classifier, such as a Support Vector Machine (SVM).
2. Fine-tune the network, by re-training some or all the layers.

(Razavian et al., 2014) showed the first possibility applied to images of different domains. Using features extracted from the OverFeat network (Sermanet et al., 2013) pre-trained on the ImageNet database, they analysed how the results can be used for several computer vision recognition tasks. Experimenting with visual classification (image classification, fine-grained recognition, attribute detection), they have obtained some excellent results. In particular, they showed the difference between CNN combined with SVM, either with and without data augmentation for all the experiments, and they compared with several methods from the previous state-of-the-art, obtaining a substantial improvement.

Also (Donahue et al., 2014) highlighted the use of CNNs for feature extraction. They used the neural network architecture of (Krizhevsky et al., 2012), and tested it on several recognition tasks, such as scene and subcategory recognition. In both cases the results were better than the ones of the state-of-the-art.

Similarly to the previously mentioned studies, (Hertel et al., 2015) addressed the drawback of the time needed to train a deep CNN. In order to accomplish this, they first trained a deep CNN on the ILSVRC-12 large dataset from an ImageNet competition, and fine-tuned it on three datasets: MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009). The MNIST dataset contains grayscale images of handwritten digits while CIFAR-10 and CIFAR-100 contain small color images of natural objects. By maintaining the learned convolution kernels and retraining only the classification part on different datasets, they obtained an accuracy rate comparable to a full training approach, suggesting that CNNs are able to learn generic feature extractors that can be used for different tasks.

(Nogueira et al., 2016) used fingerprint liveness detection datasets to fine-tune two different CNNs: CNN-Alexnet (similar to AlexNet (Krizhevsky et al., 2012)), and CNN-VGG (Simonyan and Zisserman, 2014). The obtained results show that pre-trained CNNs can detect whether a fingerprint is false or real with state-of-the-art performance, even though no task-specific techniques are used. They also reported that training a classifier with multiple datasets improves accuracy and robustness. This, in turn, suggests that combining different datasets can avoid fine tuning of hyper-parameters. For our work, the main contribution of this study is that pre-trained networks on natural images can be successfully used in fingerprint domain.

3 DATABASE

In this work the NIST Special Database 4 (NIST SD4) has been used for the experiments. It is the most important benchmark for the fingerprint classification problem. It is a dataset of 4.000 8-bit grayscale fingerprint images provided by the National Institute of Standards and Technology and collected by (Watson and Wilson, 1992a). It contains 512×512 images, classified in five classes, Arch (A), Tented arch (T), Left loop (L), Right loop (L), and Whorl (W), with 400 image pairs per class. The two images of each pair are two different rollings of the same finger.

4 METHODOLOGY

The architecture of the pre-trained CNNs are shown in Table 1. The two networks have a 224×224 fingerprint image as input, with the average fingerprint image of the training set subtracted. Both of them have 19 layers, with five convolutional and three max-pooling ones. A rectified linear unit (ReLU) is used as a non linear activation function. At the end, a softmax layer is used to obtain a probability distribution, used as confidence values of the classifier.

The fine-tuning of the CNNs was performed after the replacement of the last fully connected layer, with one that has a four-element vector, a dimensionality which is equal to the number of classes of our problem. The weights of this layer were initialised from a Gaussian distribution. The learning followed a procedure similar to other works in the literature, like (Chatfield et al., 2014), with the adoption of gradient descent with momentum. The networks were re-training for 140 epochs, where the images of the training set were previously flipped. Since fingerprint

Table 1: VGG-F and VGG-S architectures.

	VGG-F									
layer	0	1	2	3	4	5	6	7	8	9
type	input	conv	relu	pool	conv	relu	pool	conv	relu	conv
support	-	11	1	3	5	1	3	3	1	3
filt dim	-	3	-	-	64	-	-	256	-	256
num filts	-	64	-	-	256	-	-	256	-	256
stride	-	4	1	2	1	1	2	1	1	1
pad	-	0	0	0×1×0×1	2	0	0	1	0	1
layer	10	11	12	13	14	15	16	17	18	19
type	relu	conv	relu	pool	conv	relu	conv	relu	conv	softMax
support	1	3	1	3	6	1	1	1	1	-
filt dim	-	256	-	-	256	-	4096	-	4096	-
num filts	-	256	-	-	4096	-	4096	-	1000	-
stride	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	0	0	0	0	0	0	0

	VGG-S									
layer	0	1	2	3	4	5	6	7	8	9
type	input	conv	relu	pool	conv	relu	pool	conv	relu	conv
support	-	7	1	3	5	1	2	3	1	3
filt dim	-	3	-	-	96	-	-	256	-	512
num filts	-	96	-	-	256	-	-	512	-	512
stride	-	2	1	2	1	1	2	1	1	1
pad	-	0	0	0×2×0×2	0	0	0×1×0×1	1	0	1
layer	10	11	12	13	14	15	16	17	18	19
type	relu	conv	relu	pool	conv	relu	conv	relu	conv	softMax
support	1	3	1	3	6	1	1	1	1	-
filt dim	-	512	-	-	512	-	4096	-	4096	-
num filts	-	512	-	-	4096	-	4096	-	1000	-
stride	1	1	1	3	1	1	1	1	1	1
pad	0	1	0	0×1×0×1	0	0	0	0	0	0

images are not flip invariant, in case of right and left loops we changed the label of the images accordingly, so a right-loop fingerprint was assigned to the left-loop class when flipped and viceversa. No change of class is required for whorls and arches. In this way, the number of input images was doubled. During training some data augmentation techniques were adopted (Figure 2). In particular the images were randomly rotated between -20° and +20°, stretched and/or shrunk horizontally and vertically, and modified by the application of a Gaussian noise and a gray level mapping. After the augmentation, the 512×512 images were cropped in the central 400×400 part and resized in order to have the suitable dimensions for the input of the network.

During the fine-tuning, the weights of a layer are updated using a local learning rate multiplied by the global learning rate (shared with all the layers). We set the local learning rates for the earlier layers, which learn more generic features, close to 0, and the initial global learning rate to 0.05. Then, we decreased the learning rate to 0.02 after 10 epochs, and again to 0.01 after 20 epochs. Later we reduced the learning rate by a factor of 10 every time that the validation error stop decreasing. In total, five different learning rates were used. Moreover we set a momentum value to 0.9, a weight decay value to 0.0001, and a batch size value to 100. In order to accelerate the training we added batch normalisation layers. As shown by (Ioffe and Szegedy, 2015), this approach allows to use higher learning rates and in some cases it acts as a regulariser. The reason why batch normalisation helps is that inside a deep network the weights adjust the data that flows through it so that some values may differ a lot from each other. This issue, known as internal covariate shift, is solved by normalising the data after each convolutional layer. We also added two dropout layers (Srivastava et al., 2014) to reduce overfit. The

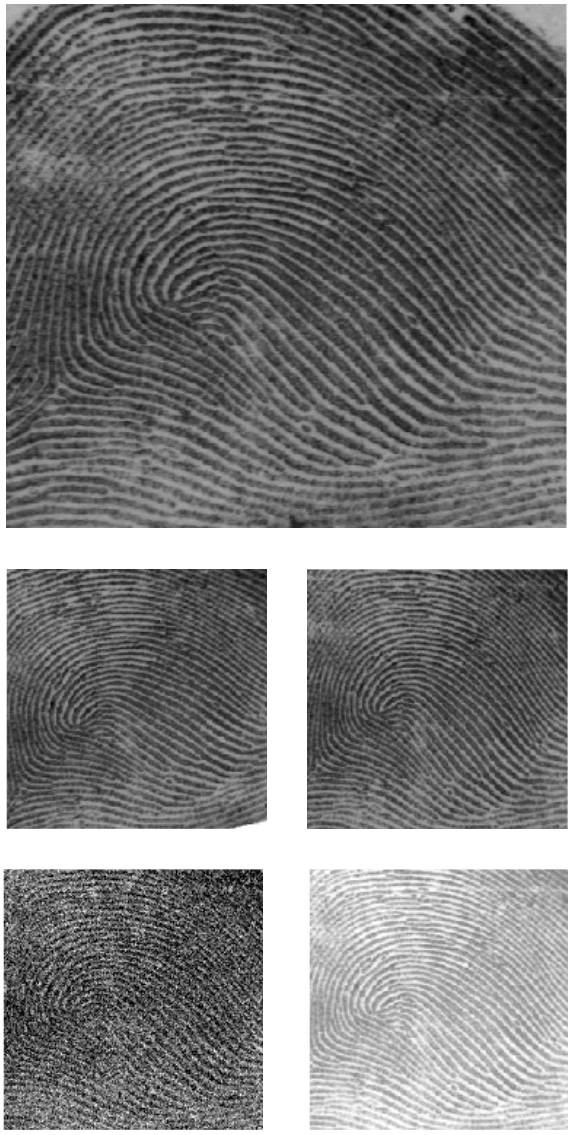

Figure 2: Data augmentation techniques. On the top the original image is shown. On the bottom we have the same image after the cropping of the central area and the application of: a rotation (top-left); a stretching and shrinking (top-right); a gaussian noise (bottom-left); a grey level mapping (bottom-right).

idea in this case is to inhibit some nodes of the network with a probability p (equals to 0.5 in our case) during training.

As mentioned before, the database we used is NIST SD4. Knowing that fingerprints are not evenly distributed among the classes, only less than 10% of them belong to arch and tented arch, and that distinguishing between these two classes is difficult because of their similarity, we chose to merge them into one category, as done in other works from the literature. Therefore, we have a dataset of four categories (arch, left loop, right loop, and whorl).

For the evaluation of the performance, we used a protocol that can be considered standard, since it is used in almost all the researches. The database, containing 4.000 fingerprint images numbered from f0001 to f2000 and from s0001 to s2000, was split into two sets: training set (from f0001 to f1000 and s0001 to s1000), and test set. Since 17.5% of the fingerprints are considered ambiguous, they have two classes assigned. For training, only one label is used, while for testing all the two are. We also decided to split the training set, choosing 100 different fingerprints as our validation set.

5 RESULTS AND DISCUSSION

For the experiments we used MATLAB and MatConvNet (Vedaldi and Lenc, 2015) combined with the NVIDIA CuDNN libraries. The models have been fine-tuned on the NVIDIA GTX 950M GPU.

Using the data augmentation techniques and the parameters described in the previous section, we fine-tuned the networks for 140 epochs, and in both cases a training error of around 8% has been reached, as shown in Figure 3 for VGG-S. The whole fine-tuning process took around 9 hour for VGG-F and around 30 hours for VGG-S on our machine, which is an Asus K550JX.

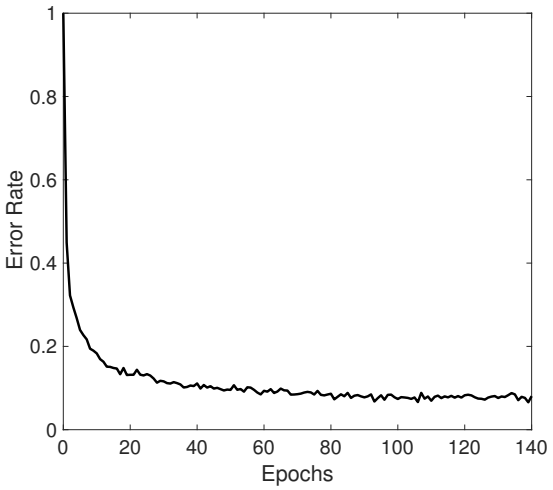

Figure 3: Fine tuning of the VGG-S network over 140 epochs.

VGG-F allows us to achieve a 94.4% of accuracy, with a testing time of 39 ms per image. The confusion matrix is shown in Table 2, and it can be seen that the network has problems in classifying right-loop and

left-loop fingerprints, with a misclassification rate of 11.25% and 7.33% respectively.

Table 2: Confusion matrix when using VGG-F.

	Assigned class				
True class	A	L	R	W	Accuracy
A	794	13	10	0	97.18%
L	23	354	3	2	92.67%
R	41	1	355	3	88.75%
W	3	5	8	385	96.01%

On the other hand, using VGG-S a 95.05% of accuracy was achieved, with a testing time of 77 ms per image. Also in this case the network shows problems in the classification of right-loop and left-loop fingerprints, where the misclassification rate is 9.39% and 8.14% respectively (Table 3). Some examples of misclassified fingerprints can be seen in Figure 4.

Table 3: Confusion matrix when using VGG-S.

	Assigned class				
True class	A	L	R	W	Accuracy
A	809	8	7	0	98.18%
L	27	350	2	2	91.86%
R	31	0	357	6	90.61%
W	3	5	8	385	96.01%

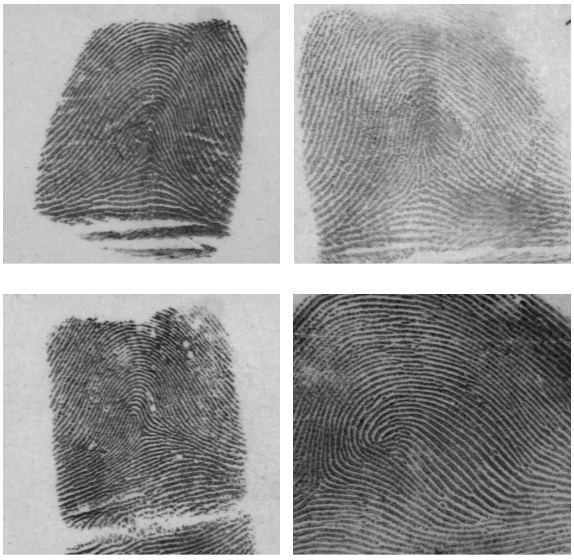

Figure 4: Misclassified samples in NIST SD4 using VGG-S. Top-left: arch classified as left loop. Top-right: arch classified as right loop. Bottom-left: left loop classified as arch. Bottom-right: right loop classified as arch.

Sometimes, the differences between two classes are hard to determine, even for a human expert. For this reason a rejection option can be applied: if a fingerprint is hard to classify by the system, then it is rejected, and it will be evaluated by an expert. We chose to apply a threshold to the confidence score of the best class chosen by our classifier: if the score is less than this threshold, then the fingerprint is rejected. Figure 5 shows the improvements of the accuracy of our classifier based on VGG-S if a rejection option is applied.

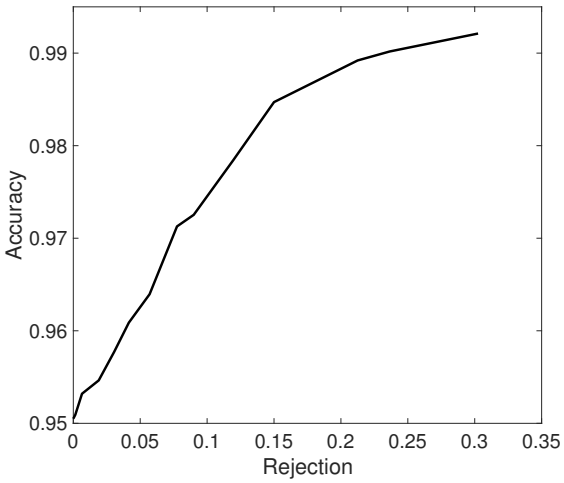

Figure 5: Accuracy versus rejection curve for our classifier based on VGG-S.

Table 4 compares our results with similar works in the literature. We reported also the work of (Candela et al., 1995) because it is considered a milestone (Maltoni et al., 2009), since the introduced system, PCASYS, is open source and it was one of the first studies whose results were reported on publicly available databases. We can see that the achieved error rate makes our method competitive, even though more work should be done to get closer to the performance reported by (Cao et al., 2013). However, the main advantage of our technique is that it is not based on any of the standard features for classification, such as orientation image (Cappelli et al., 1999), (Cappelli et al., 2003), (Cappelli and Maio, 2004), (Park and Park, 2005), (Tan et al., 2005), singular point (Li et al., 2008), and ridge line flow (Candela et al., 1995). This means that, apart from a crop of the image, no preprocessing is needed at test time, keeping the computational complexity of our method low. As mentioned before, we are able to classify a fingerprint image in 39 ms or 77 ms using VGG-F or VGG-S respectively. This confirms that feature-learning approaches, like the one of (Tan et al., 2005) that reported an average run-time for one fingerprint test of 71 ms on a SUN Ultra II workstation with a 200MHZ CPU, are faster. For example, (Cao et al., 2013) used an algorithm where they classify the fingerprints based on

their orientation image. They obtain an average orientation extraction time of 880 ms and an average classification time of 3.43 s on a 3.4 GHz Intel Pentium 4 processor.

Table 4: Error rates of different classification methods on NIST SD4.

Method	4 classes
(Candela et al., 1995)	11.4%
(Cappelli et al., 1999)	5.5%
(Cappelli et al., 2003)	3.7%
(Cappelli and Maio, 2004)	4.7%
(Zhang and Yan, 2004)	7.5%
(Park and Park, 2005)	6.0%
(Tan et al., 2005)	6.7%
(Li et al., 2008)	5.0%
(Cao et al., 2013)	2.8%
Our Method - VGG-F	5.6%
Our Method - VGG-S	4.95%

6 CONCLUSION

In this paper, two pre-trained CNNs, VGG-F and VGG-S, have been used to address the fingerprint classification problem. The results show that the performance obtained with our approach are close to the state-of-the-art, with an accuracy rate of 94.4% and 95.05% when using VGG-F and VGG-S respectively. This confirms that transfer learning can be used to achieve high accuracy in fingerprint classification. The main advantage of our approach is that it does not require a heavy preprocessing stage, as in the other related works, where some features, such as the orientation image, have to be extracted. In our case, when a fingerprint image is provided to the trained CNN, it extracts a set of features with the filters learned during the training, and classifies it.

REFERENCES

Candela, G. T., Grother, P. J., Watson, C. I., Wilkinson, R., and Wilson, C. L. (1995). PCASYS - A pattern-level classification automation system for fingerprints. *NIST technical report NISTIR*, 5647.

Cao, K., Pang, L., Liang, J., and Tian, J. (2013). Fingerprint classification by a hierarchical classifier. *Pattern Recognition*, 46(12):3186–3197.

Cappelli, R. and Maio, D. (2004). The state of the art in fingerprint classification. In *Automatic Fingerprint Recognition Systems*, pages 183–205. Springer.

Cappelli, R., Maio, D., and Maltoni, D. (1999). Fingerprint classification based on multi-space KL. In *Proceedings Workshop on Automatic Identification Advances Technologies (AutoID99)*, pages 117–120.

Cappelli, R., Maio, D., Maltoni, D., and Nanni, L. (2003). A two-stage fingerprint classification system. In *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pages 95–99. ACM.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of British Machine Vision Conference*, pages 1–11.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. 32:647–655.

Galar, M., Derrac, J., Peralta, D., Triguero, I., Paternain, D., Lopez-Molina, C., García, S., Benítez, J. M., Pagola, M., Barrenechea, E., et al. (2015). A survey of fingerprint classification part I: Taxonomies on feature extraction methods and learning models. *Knowledge-based systems*, 81:76–97.

Hertel, L., Barth, E., Kaster, T., and Martinetz, T. (2015). Deep convolutional neural networks as generic feature extractors. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–4. IEEE.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. 37:448–456.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Cortes, C., and Burges, C. J. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist. Accessed: 01-03-2016.

Li, J., Yau, W.-Y., and Wang, H. (2008). Combining singular points and orientation image information for fingerprint classification. *Pattern Recognition*, 41(1):353–366.

Maltoni, D., Maio, D., Jain, A., and Prabhakar, S. (2009). *Handbook of fingerprint recognition*. Springer Science & Business Media.

Mayhew, S. (2015). History of biometrics. http://www.biometricupdate.com/201501/history-of-biometrics. 01/09/2016.

Nogueira, R., Lotufo, R., and Campos Machado, R. (2016). Fingerprint liveness detection using convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 11(6):1206–1213.

Park, C. H. and Park, H. (2005). Fingerprint classification using fast Fourier transform and nonlinear discriminant analysis. *Pattern Recognition*, 38(4):495–503.

Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). OverFeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Tan, X., Bhanu, B., and Lin, Y. (2003). Learning features for fingerprint classification. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pages 318–326. Springer.

Tan, X., Bhanu, B., and Lin, Y. (2005). Fingerprint classification based on learned features. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35(3):287–300.

Vedaldi, A. and Lenc, K. (2015). MatConvNet: Convolutional neural networks for MATLAB. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 689–692. ACM.

Watson, C. I. and Wilson, C. L. (1992a). NIST Special Database 4. https://www.nist.gov/srd/nist-special-database-4. Accessed: 01-09-2016.

Watson, C. I. and Wilson, C. L. (1992b). NIST Special Database 9. https://www.nist.gov/srd/nist-special-database-9. Accessed: 01-09-2016.

Watson, C. I. and Wilson, C. L. (1993). NIST Special Database 14. https://www.nist.gov/srd/nist-special-database-14. Accessed: 01-09-2016.

Zhang, Q. and Yan, H. (2004). Fingerprint classification based on extraction and analysis of singularities and pseudo ridges. *Pattern Recognition*, 37(11):2233–2243.